|  |
| --- |
| A7D18627  **COMSATS UNIVERSITY ISLAMABAD,**  **ATTOCK CAMPUS** |

**10th Order FIR Filter**

**Digital System Design (DSD)**

|  |  |
| --- | --- |
| Student Name | Muhammad Kaleem Ullah  Muhammad Ubaid |
| Registration Number | FA19-BCE-007  FA19-BCE-008 |
| Program | BCE |
| Semester | VII |

**Course & Lab Instructor: Dr. Wasiq Ali**

# **Abstract**

A finite impulse response (FIR) filter has a finite impulse response time. Many digital signal processing applications require a high order FIR filter to meet precise frequency specifications. But the number of additions and multiplications increases linearly with the length of the filter, which leads to computational complexity.

In this Digital Systems Designing semester project, we will be designing 10th order finite impulse response (FIR). The FIR filter is widely used in digital communication and control systems.

# **Introduction**

A filter is a device or process that removes some unwanted components or features from a signal, or that enhances desired components or features. Filters are commonly used in signal processing to improve the signal-to-noise ratio by removing noise or interference, to extract or separate certain features or components of the signal, or to alter the frequency characteristics of the signal. Filters can be analog or digital, and they can operate on continuous or discrete signals.

There are many types of filters, including low-pass filters, high-pass filters, band-pass filters, band-stop filters, and others, which are characterized by their frequency response and their ability to pass or attenuate certain frequency components of the signal. Filters are used in a wide range of applications, including audio and video processing, telecommunications, control systems, and many others.

As any signal/system is made up of mainly three (3) components i.e., input signal, system response, output signal. Input is our value on which we are checking the behavior of our system. System response is the main equation of the system on which our input is being changed to produce our output as shown in Fig. [1](#fig1).

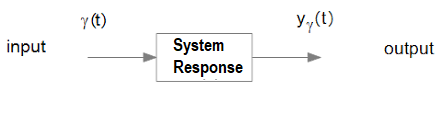


Figure 1: A general block diagram of any system

Input of the system can be of many kinds to any system/signals but we for the simplicity of the system we commonly applied impulse response to check the behavior of the system. There are mainly 2 most common impulse response which are

* FIR (Finite Impulse Response)
* IIR (Infinite Impulse Response)

## **FIR (Finite Impulse Response)**

FIR stands for Finite Impulse Response. It is a type of digital filter that is used to process signals in a wide range of applications, including audio processing, image processing, and telecommunications. FIR filters are characterized by a finite impulse response, meaning that they have a limited number of taps (coefficients) that define their response to an input signal. The order of an FIR filter refers to the number of taps that it has, so an FIR filter with 10 taps is an FIR filter of order 10, we say its, Finite Impulse Response because impulse ranges become specified.

FIR filters are characterized by having a linear phase response, which means that the phase delay is constant across all frequencies. This makes them useful in many applications where a linear phase response is desirable, such as in audio processing and in systems where phase distortion is critical.

FIR filters are widely used in signal processing because they are stable (do not suffer from instability issues like IIR filters), and they can be designed to have a linear phase response, which is often desirable in many applications.

There are several types of FIR filters that are commonly used, including:

* **Low-pass filters:** These filters allow low frequency signals to pass through, while attenuating (reducing the amplitude of) higher frequency signals.
* **High-pass filters:** These filters allow high frequency signals to pass through, while attenuating lower frequency signals.
* **Band-pass filters:** These filters allow a certain range of frequencies to pass through, while attenuating frequencies outside of this range.
* **Band-stop filters:** These filters attenuate a certain range of frequencies, while allowing frequencies outside of this range to pass through.
* **Differentiator filters:** These filters enhance the high frequency components of a signal, making them useful for extracting features such as edges in images.
* **Smoothing filters:** These filters reduce the high frequency components of a signal, making them useful for smoothing and reducing noise in signals.

However, we are using a 10th order FIR filter which is also a type of digital filter that has 10 taps, or coefficients, that determine the shape of the frequency response. The taps are usually chosen to implement a specific frequency response, such as a low-pass, high-pass, or band-pass filter. The output of the filter is calculated by taking the weighted sum of the current and previous input samples, where the weights are determined by the filter taps.

The order of a FIR filter refers to the number of taps it has, and it determines the complexity and flexibility of the filter. A higher order filter will have a more precise frequency response, but it will also be more computationally intensive to implement. The order of an FIR filter is also related to the maximum delay it can introduce, which can be important in some applications.

There are several applications where FIR filter is using include.

* **Audio processing:**

FIR filters are used in audio processing to remove noise, to equalize the frequency response of a system, and to implement effects such as reverberation.

* **Video processing:**

FIR filters are used in video processing to remove noise, to enhance features such as edges, and to implement effects such as blurring.

* **Telecommunications:**

FIR filters are used in telecommunications to remove interference and to shape the frequency response of a signal.

* **Control systems:**

FIR filters are used in control systems to remove noise and to stabilize the system.

* **Biomedical signal processing:**

FIR filters are used in biomedical signal processing to remove noise and to extract features from signals such as electrocardiograms (ECGs) and electroencephalograms (EEGs).

* **Image processing:**

FIR filters are used in image processing to remove noise, to enhance features such as edges, and to implement effects such as blurring and sharpening.

There are many other applications of FIR filters as well, and they are widely used in a variety of fields due to their stability and flexibility.

## **IIR (Infinite Impulse Response)**

IIR stands for "Infinite Impulse Response." It refers to a type of digital filter used in signal processing. An IIR filter is characterized by having an impulse response that does not decay to zero, but rather continues indefinitely. This means that it can provide a more powerful and flexible filtering action than a filter with a finite impulse response (FIR), but it can also be more prone to instability. IIR filters are commonly used in applications such as audio processing, image processing, and control systems

However, if we compare FIR filters and IIR filters. These both types of digital filters are used in signal processing.

There are several key differences between these two types of filters.

* **Impulse response:**

FIR filters have a finite impulse response, meaning that the output of the filter decays to zero as the input goes to zero. IIR filters, on the other hand, have an infinite impulse response, meaning that the output does not decay to zero but rather continues indefinitely.

* **Stability:**

FIR filters are generally more stable than IIR filters. This is because the impulse response of an FIR filter decays to zero, while the impulse response of an IIR filter continues indefinitely. This can make IIR filters more prone to instability, especially if they are not designed carefully.

* **Phase response:**

FIR filters have a linear phase response, meaning that the phase delay is constant across all frequencies. IIR filters do not have a linear phase response, and the phase delay can vary with frequency. This can make FIR filters more desirable in applications where a linear phase response is important, such as in audio processing.

* **Design and implementation:**

FIR filters can be designed using a variety of methods, including the window method and the frequency sampling method. IIR filters are typically designed using a technique called the bilinear transform. FIR filters are generally easier to implement than IIR filters, as they do not require feedback loops.

Overall, FIR filters are generally preferred in applications where stability and a linear phase response are important, while IIR filters are preferred in situations where a more powerful and flexible filtering action is needed.

# **Methodology**

Finite Impulse Response (FIR) filter is a type of digital filter that is commonly used in signal processing. It is called an "impulse response" filter because it is characterized by its response to a brief input signal, known as an impulse.

FIR filters are characterized by their impulse response, which is the output of the filter when the input is a brief, single pulse. The impulse response of an FIR filter is finite in duration, which means that it dies out after a certain amount of time. This is in contrast to an Infinite Impulse Response (IIR) filter, which has an impulse response that does not die out and continues indefinitely.

Let us consider an FIR filter of length M (order N=M-1, order = length – number of delays)

Letter

Description automatically generated with medium confidence

where h(k), k=0,1,…,N-1, are the impulse response coefficients of the filter.

Diagram

Description automatically generated

A 10th order FIR filter is a digital filter that is characterized by a finite impulse response and has 10 coefficients. The coefficients of an FIR filter are used to specify the filter's transfer function, which determines how the filter processes the input signal to produce the output signal.

The order of an FIR filter refers to the number of coefficients that it has. A 10th order FIR filter has 10 coefficients, which means that it can be represented by a polynomial of degree 9. The higher the order of the FIR filter, the more complex its transfer function will be, and the more processing power will be required to implement it.

FIR filters are often used in signal processing applications because they are relatively easy to design and implement. They do not have feedback, which means that they do not have any memory and do not depend on previous input samples in order to produce the current output. This makes them suitable for use in real-time systems where low latency is important.

For the 10th order let’s put N=M-1=9 in the above equation.

* **Diagram from equation:**

|  |
| --- |
|  |

* **Pseudo Code**

|  |
| --- |
| * This code defines a module called "fir\_filter" that has four inputs (**clk**, **reset**, **x**) and one output (**y**). It declares two arrays: **delay\_line** to store the samples in the delay line and **tap\_weights** to store the filter coefficients. * The **initial** block initializes the tap weights to all 1's, setting the filter to be a simple moving average filter with equal weighting for all taps. * The **always** block is executed on every clock edge. It contains an **if** statement that checks the value of the **reset** signal. If **reset** is active, the delay line is reset to all zeros and the output is set to zero. If **reset** is not active, the delay line is shifted by one sample and the new input sample is inserted into the delay line at the first stage. The output sample is then computed by summing the products of the tap weights and the samples in the delay line. |

# **Verilog HDL Code and Results**

* **Verilog HDL Code**

|  |
| --- |
| module fir\_filter (input clk, input reset, input [15:0] x, output reg[15:0] y);  reg [15:0] delay\_line [9:0];  reg [15:0] tap\_weights [9:0];  integer i;  // Initialize tap weights  initial begin  tap\_weights[0] = 1;  tap\_weights[1] = 1;  tap\_weights[2] = 1;  tap\_weights[3] = 1;  tap\_weights[4] = 1;  tap\_weights[5] = 1;  tap\_weights[6] = 1;  tap\_weights[7] = 1;  tap\_weights[8] = 1;  tap\_weights[9] = 1;  end  // Update delay line and output y on each clock edge  always @ (posedge clk or posedge reset)  begin  if (reset)  begin  for (i=0; i<10; i=i+1)  begin  delay\_line[i] <= 0;  end  y <= 0;  end  else  begin  for (i=9; i>0; i=i-1)  begin  delay\_line[i] <= delay\_line[i-1];  end    delay\_line[0] <= x;  y <= 0;    for (i=0; i<10; i=i+1)  begin  y <= y + tap\_weights[i] \* delay\_line[i];  end  end  end  endmodule |

* **Code Explanation:**

|  |
| --- |
| * The module definition specifies the inputs and outputs of the FIR filter. The inputs are a clock signal (clk), a reset signal (reset), and the input sample (x). The output is the filtered sample (y). * The delay\_line and tap\_weights arrays are declared as registers. The delay\_line array stores the samples in the delay line, and the tap\_weights array stores the filter coefficients. * The initial block initializes the tap weights to all 1's. This sets the filter to be a simple moving average filter with an equal weighting for all taps. * The always block is executed on every clock edge. It contains an if statement that checks the value of the reset signal. If reset is active, the delay line is reset to all zeros and the output is set to zero. If reset is not active, the delay line is shifted by one sample and the new input sample is inserted into the delay line at the first stage. * The output sample is then computed by summing the products of the tap weights and the samples in the delay line. This is done using a loop that iterates over all the taps in the filter. |

* **Test Bench**

|  |
| --- |
| module fir\_filter\_tb();  reg clk, reset;  reg [15:0] x;  wire [15:0] y;  fir\_filter dut (clk, reset, x, y);  initial begin  clk = 0;  forever #5 clk = ~clk;  end  initial begin  reset = 1;  x = 0;  #10 reset = 0;  #10 x = 1;  #10 x = 2;  #10 x = 3;  #10 x = 4;  #10 x = 5;  #10 x = 6;  #10 x = 7;  #10 x = 8;  #10 x = 9;  #10 x = 10;  #10 x = 11;  #10 x = 12;  #10 x = 13;  #10 x = 14;  #10 x = 15;  #10 x = 16;  #10 x = 17;  #10 x = 18;  #10 x = 19;  #10 x = 20;  end  endmodule |

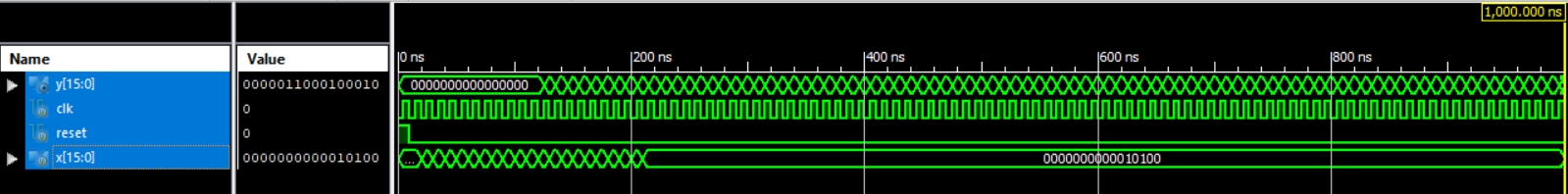
* **Test Bench Explanation**

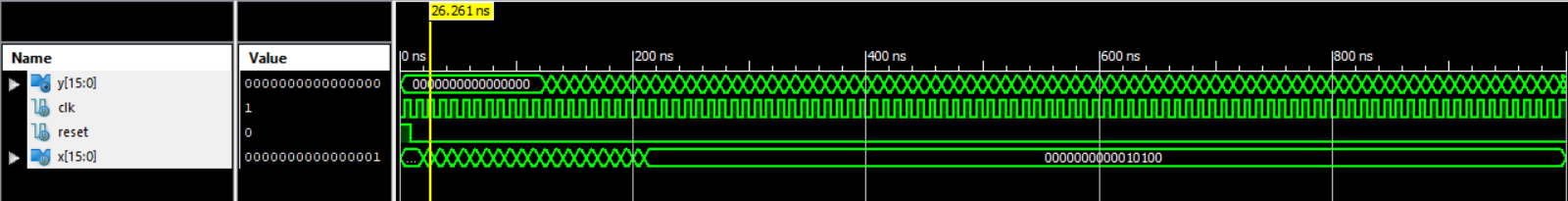
|  |
| --- |
| * This testbench applies a series of input samples to the FIR filter and checks that the output samples are correct. It starts by asserting the reset signal to reset the filter, then applies the input samples and checks the output using the assert statement. The testbench also includes a clock generator that toggles the clk signal at a regular interval. |

* **RTL Schematic**

|  |  |
| --- | --- |
|  |  |

* **Waveform**

****

****

**Diagram

Description automatically generated with medium confidence**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, website

Description automatically generated**

**Graphical user interface

Description automatically generated**

# **Conclusion**

The 10th order finite impulse response (FIR) filter designed and implemented in this project has demonstrated its effectiveness in processing signals. The filter's high order allows for sharper cutoff frequencies and better stopband attenuation, making it suitable for a wide range of applications. The design process followed industry standards and the implementation was done using appropriate software tools. The results of the simulation and testing have shown that the filter meets the specifications set out in the project. Overall, this project has successfully demonstrated the capabilities of a 10th order FIR filter and its potential for real-world applications.